SQUALL: A Controlled Natural Language as Expressive as SPARQL 1.1

Sébastien Ferré

SQUALL: a Controlled Natural Language as Expressive as SPARQL 1.1

Sébastien Ferré

IRISA, Université de Rennes 1
Campus de Beaulieu, 35042 Rennes cedex, France
Email: ferre@irisa.fr

Abstract. The Semantic Web is now made of billions of triples, which are available as Linked Open Data (LOD) or as RDF stores. The most common approach to access RDF datasets is through SPARQL, an expressive query language. However, SPARQL is difficult to learn for most users because it exhibits low-level notions of relational algebra such as union, filters, or grouping. We present SQUALL, a high-level language for querying and updating an RDF dataset. It has a strong compliance with RDF, covers all features of SPARQL 1.1, and has a controlled natural language syntax that completely abstracts from low-level notions. SQUALL is available as two web services: one for translating a SQUALL sentence to a SPARQL query or update, and another for directly querying a SPARQL endpoint such as DBpedia.

1 Introduction

An open challenge of the Semantic Web [12] is *semantic search*, i.e., the ability for users to browse and search semantic data according to their needs. Semantic search systems can be classified according to their *usability*, the *expressive power* they offer, their *compliance* to Semantic Web standards, and their *scalability*. The most expressive approach by far is to use SPARQL [17], the standard RDF query language. SPARQL 1.1[1] features graph patterns, filters, unions, differences, optionals, aggregations, expressions, subqueries, ordering, etc. However, SPARQL is also the least usable approach, as it is defined at a low-level in terms of relational algebra. There are mostly two approaches to make more usable semantic search systems: navigation and natural language. Navigation is used in *semantic browsers* (e.g., Fluidops Information Workbench[2]), and in *semantic faceted search* (e.g., SlashFacet [11], BrowseRDF [16], Sewelis [6]). Semantic faceted search can reach a significant expressiveness, but still much below than SPARQL 1.1, and it does not scale easily to large datasets such as DBpedia[3]. Natural language is used in search engines in various forms, going from full natural language (e.g., FREyA [3], Aqualog [14]) to mere keywords (e.g., NLP-Reduce [13]) through controlled natural languages (e.g., Ginseng [1]). Questions

[1] http://www.w3.org/TR/sparql11-query/
[2] http://iwb.fluidops.com/
[3] http://dbpedia.org

in natural language are translated to SPARQL queries, but in general, only a small fragment of SPARQL is used. This means that even if full natural language is allowed, expressiveness is in fact strongly limited. In practice, SPARQL remains the main way to search RDF datasets. A first reason may be that users really need expressiveness in practice, at least when they get specifically interested in a dataset. A second reason may be that most semantic search systems require a lot of preparation before being applied to a specific dataset (e.g., definition of facets or lexicon, derivation of a grammar from an ontology [1]), while SPARQL requires no preparation at all.

A less studied aspect is the update of RDF datasets, i.e., the insertion and deletion of triples. SPARQL 1.1 offers an update language to this purpose. Proposals for more usable interfaces have been made in faceted search (e.g., UTILIS [10]), and in CNL (e.g., ACE [8]). We think that update (and creation) of RDF data is as important as querying because if no data is created, there is nothing to be searched.

In this paper, we present SQUALL, a Semantic Query and Update High-Level Language[4]. Its contribution is to offer an expressiveness that is equivalent to SPARQL 1.1 Query/Update (SPARQL for short), while providing a high-level syntax that completely abstracts from low-level notions such as bindings or relational algebra. SQUALL can be translated to SPARQL, and is therefore fully compliant with Semantic Web standards. In fact, SQUALL qualifies as a Controlled Natural Language (CNL) [19,7]. The main advantage of CNLs is to reuse the cognitive capabilities of people for communicating knowledge, and therefore to reduce the learning effort for using the language. To the best of our knowledge, no existing CNL is strongly compliant with RDF and SPARQL. ACE [7] has its own underlying formalism (Discourse Representation Constructs), and SOS and Rabbit cover OWL ontologies and assume linguistic knowledge [18]. SQUALL does not require any domain-specific linguistic knowledge: e.g., knowing that "person" is a noun, whose plural is "people", and that "knows" is a transitive verb, whose passive is "known". This means that SQUALL is less natural at the lexical level, but that it is applicable to SPARQL endpoints without any preparation.

Section 2 is a short introduction to the Semantic Web and SPARQL. Section 3 describes the different steps that enables the translation from SQUALL sentences to SPARQL queries and updates. Section 4 evaluates the expressiveness of SQUALL by giving for each SPARQL feature its counterpart in SQUALL along with examples. Section 5 evaluates the naturalness of SQUALL on the QALD benchmark. Section 6 concludes the paper.

2 Semantic Web: RDF and SPARQL

The Semantic Web (SW) is founded on several representation languages, such as RDF, RDFS, and OWL, which provide increasing inference capabilities [12].

[4] Web forms, examples, and source code can be found from the SQUALL homepage: `http://www.irisa.fr/LIS/softwares/squall`.

The two basic units of these languages are *resources* and *triples*. A resource can be either a URI (Uniform Resource Identifier), a literal (e.g., a string, a number, a date), or a *blank node*, i.e., an anonymous resource. A URI is the absolute name of a *resource*, i.e., an entity, and plays the same role as a URL w.r.t. web pages. Like URLs, a URI can be a long and cumbersome string (e.g., `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`), so that it is often denoted by a qualified name (e.g., `rdf:type`), where `rdf:` is the RDF namespace. In the N3 notation, the default namespace `:` can be omitted for qualified names that do not collide with reserved keywords (*bare qualified names*).

A triple (s p o) is made of 3 resources, and can be read as a simple sentence, where s is the subject, p is the verb (called the predicate), and o is the object. For instance, the triple (`Bob knows Alice`) says that "Bob knows Alice", where `Bob` and `Alice` are the bare qualified names of two individuals, and `knows` is the bare qualified name of a property, i.e., a binary relation. The triple (`Bob rdf:type man`) says that "Bob has type man", or simply "Bob is a man". Here, the resource `man` is used as a class, and `rdf:type` is a property from the RDF namespace. The triple (`man rdfs:subClassOf person`) says that "man is a subclass of person", or simply "every man is a person". The set of all triples of a knowledge base forms an RDF graph.

Query languages provide on semantic web knowledge bases the same service as SQL on relational databases. They generally assume that implicit triples have been inferred and added to the base. The standard RDF query language, SPARQL, reuses the `SELECT FROM WHERE` shape of SQL queries, using graph patterns in the `WHERE` clause. A graph pattern G is one of:

- a triple pattern (s p o .) made of RDF terms and variables (e.g., `?x`),
- a join of two patterns ($\{G_1\ G_2\}$),
- an union of two patterns (G_1 `UNION` G_2),
- an optional pattern (`OPTIONAL` G_1),
- a filter pattern (`FILTER` C), where C is a constraint, i.e., either a Boolean expression based on primitive predicates (e.g., comparison, string matching), or a negated graph pattern (`NOT EXISTS` G_1),
- a named graph pattern (`GRAPH` g G_1), where g is URI or a variable denoting a named graph,
- a subquery.

Aggregations and expressions can be used in the `SELECT` clause (e.g., `COUNT`, `SUM`, `2 * ?x`), and `GROUP BY` clauses can be added to a query. Solution modifiers can also be added to the query for ordering results (`ORDER BY`) or returning a subset of results (`OFFSET`, `LIMIT`). Other query forms allow for closed questions (`ASK`), for returning the description of a resource (`DESCRIBE`), or for returning RDF graphs as results instead of tables (`CONSTRUCT`). SPARQL has been extended into an update language to insert and delete triples in/from a graph. The most general update form is `DELETE` D `INSERT` I `WHERE` G, where I and D must be sets of triple patterns, and G is a graph pattern that defines bindings for variables occuring in I and D.

3 Translation from SQUALL to SPARQL

The idea behind SQUALL is to offer a high-level substitute of SPARQL. This implies that all of SQUALL should be translatable to SPARQL, and that all of SPARQL should be expressible in SQUALL. We do not have yet a formal proof of the latter, but all features of SPARQL are covered in SQUALL, up to a few minor exceptions, as shown in Section 4. The implementation of SQUALL as a translator to SPARQL is an obvious choice as it leverages existing work on efficient SPARQL query engines, and satisfies interoperability with existing RDF stores, which provide access through SPARQL endpoints. We now describe the different steps of such a translation.

3.1 Lexical Analysis

In the current implementation of SQUALL, there are no proper lexical analysis because we use the same lexical conventions as in SPARQL, plus bare qualified names like in N3 (see Section 2). This comes from our choice to make SQUALL directly applicable to RDF datasets without preparation. Of course, if linguistic knowledge is available for the resources of the datasets (e.g., "actor", "stars", "starring" all refer to the property `dbpedia:starring`), then a preprocessing stage may be applied on SQUALL sentences to allow for more natural sentences at the lexical level. Fortunately, namespaces and bare qualified names allow for relatively natural sentences, as shown in examples in this paper and on the Web page. For example, DBpedia uses three namespaces: one for the ontology (classes and properties), another for additional properties, and a last one for individual resources. By associating to them respectively the prefixes :, `dbp:`, `res:`, we can write in SQUALL "Which Film has director res:Tim_Burton ?". `Film` stands for the URI `http://dbpedia.org/ontology/Film`, and `res:Tim_Burton` stands for the URI `http://dbpedia.org/resource/Tim_Burton`. In SQUALL, classes can be used as nouns and intransitive verbs, properties can be used as relation nouns and transitive verbs, and resources can be used as proper nouns.

3.2 Syntactic and Semantic Analysis

The syntactic and semantic analysis of SQUALL are formally defined and implemented as a Montague grammar made of around 100 rules[5]. Montague grammars [4] are an approach to natural language semantics that is based on formal logic and λ-calculus. It is named after the American logician Richard Montague, who pioneered this approach [15]. A Montague grammar is a context-free generative grammar, where each rule is decorated by a λ-term that denotes the semantics of the syntactic construct defined by the rule. The semantics is defined in a fully compositional style, i.e., the semantics of a construct is always

[5] The full Montague grammar can be found in the source code at `https://bitbucket.org/sebferre/squall2sparql/src` (file `syntax.ml`), or in a previous paper [5] for an earlier version of SQUALL.

a composition of the semantics of sub-constructs. The obtained semantics for a valid SQUALL sentence serves as an intermediate representation before generation to possibly different target languages, here SPARQL.

SQUALL sentences are decomposed into noun phrases, verb phrases, relatives, determiners, prepositional phrases. They can express assertions when ending with a full stop (e.g., "res:Paris is the capital of res:France.") or questions when ending with a question mark (e.g., "What is the capital of res:France?"). So far, anaphoras are handled with variables (e.g., ?X), but those are rarely needed. As an illustration, we consider a complex sentence that covers many features of SQUALL: "For which researcher-s ?X, in graph DBLP every publication whose author is ?X and whose year is greater than 2000 has at least 2 author-s?". Its syntactic analysis is (see [5] for details)

> "[$_S$for [$_{NP}$[$_{Det}$which] [$_{NG1}$[$_{P1}$researcher-s] [$_{AR}$[$_{App}$?X]]]], [$_S$[$_{PP}$in [$_{Prep}$graph] [$_{NP}$DBLP]] [$_S$[$_{NP}$[$_{Det}$every] [$_{NG1}$[$_{P1}$publication] [$_{AR}$[$_{Rel}$[$_{Rel}$whose [$_{NG2}$[$_{P2}$author]] [$_{VP}$is [$_{NP}$?X]]] and [$_{Rel}$whose [$_{NG2}$[$_{P2}$year]] [$_{VP}$is [$_{Rel}$greater than [$_{NP}$2000]]]]]] [$_{VP}$has [$_{Det}$at least 2] [$_{P2}$author-s]]]]]]".

3.3 SPARQL Generation

The last step is the generation of a SPARQL query or update from the intermediate representation of semantics. It is much simpler than syntactic and semantic analysis (around 100 lines of code) because it mostly consists in mapping logical constructs to SPARQL constructs, which are at the same level of abstraction. Note that the intermediate representation makes it easy to support another target query language, e.g., Datalog [2]. As an illustration, the SPARQL translation of the above example is as follows:

```
SELECT DISTINCT ?X WHERE {
    ?X a :researcher .
    FILTER NOT EXISTS {
        GRAPH :DBLP {
            ?x3 a :publication .
            ?x3 :author ?X .
            ?x3 :year ?x6 .
            FILTER (?x6 > 2000) . }
        FILTER NOT EXISTS {
            GRAPH :DBLP {
                { SELECT DISTINCT ?x3 (COUNT(?x9) AS ?x7)
                  WHERE { ?x3 :author ?x9 . }
                  GROUP BY ?x3 }
                FILTER (2 <= ?x7) . } } } }
```

The two nested `FILTER NOT EXISTS` encode the universal quantifier "every", and the subquery with aggregation encodes the numeric quantifier "at least 2".

3.4 Implementation as a Web Service

SQUALL is available as two Web services. A *translation form* takes a SQUALL sentence and returns its SPARQL translation. A *query form* takes a SPARQL

endpoint URL, namespace definitions, and a SQUALL sentence, sends the SPARQL translation to the endpoint, which returns the list of answers to the query. SQUALL is also available as a command line tool that can be called from scripts or programs locally. SQUALL is implemented in about 2000 lines of OCaml[6], a functional language where Montague grammars have a natural encoding. The source code is available as a BitBucket repository from the SQUALL homepage.

4 Expressiveness Compared to SPARQL

We evaluate the expressiveness of SQUALL by giving for each SPARQL feature its counterpart in SQUALL. This list of features is adapted and extended from a comparison of RDF query languages [9]. For each feature, SQUALL sentences are given as illustrations. For the sake of simplicity, we assume that all resources belong to a same namespace so that bare qualified names can be used (e.g., "person", "author", "NLDB"). The SPARQL translation of SQUALL sentences can be obtained from the translation form at `http://lisfs2008.irisa.fr/ocsigen/squall/`.

Triple patterns. Each noun or non-auxiliary verb plays the role of a class or a predicate in a triple pattern. If a question is about a class or a predicate, the verbs "belongs" and "relates" are respectively used.

- "Which person is the author of a publication whose publication_year is 2012?"
- "To which nationality does John_Smith belong?" (here, "nationality" is a meta-class whose instances are classes of persons: e.g., "French", "German").
- "What relates John_Smith to Mary_Well?"

Updates. Updates are obtained by sentences in the affirmative. A sequence of affirmative sentences generates a sequence of updates.

- "Paper42 has author John_Smith and has publication_year 2012."
- "John_Smith know-s Mary_Well. Mary_Well know-s John_Smith."

Queries. `SELECT` queries are obtained by open questions, using one or several question words ("which" as a determiner, "what" or "who" as a noun phrase). Queries with a single selected variable can also be expressed as imperative sentences. `ASK` queries are obtained by closed questions, using either the word "whether" in front of an affirmative sentence, or using auxiliary verbs and subject-auxiliary inversion.

- "Which person is the author of which publication?"
- "Give me the author-s of Paper42."
- "Whether John_Smith know-s Mary_Well?"
- "Does Mary_Well know the author of Paper42?"

[6] `http://caml.inria.fr/ocaml/`

Solution modifiers. The ordering of results (`ORDER BY`) and partial results (`LIMIT`, `OFFSET`) are expressed with adjectives like "highest", "2nd lowest", "10 greatest".

- "Which person-s have the 10 greatest age-s?"
- "What are the author-s of the publication-s whose publication_year is the 2nd latest?"

Built-ins. Built-in functions and operators used in SPARQL filters and expressions are expressed by pre-defined nouns, verbs, and relational adjectives: e.g., "month", "contains", "greater than". They can therefore be used like classes and properties.

- "Which person has a birth_date whose month is 3 and whose year is greater than 2000?"
- "Give me the publication-s whose title contains "natural language"?"

Join. The coordination "and" can be used with all kinds of phrases. It generates complex joins at the relational algebra level.

- "John_Smith and Mary_Well have age 42 and are an author of Paper42 and Paper43."

Union. Unions of graph patterns are expressed by the coordination "or", which can be used with all kinds of phrases, like "and".

- "Which teacher or student teach-es or attend-s a course whose topic is NL or DB?"

Option. Optional graph patterns are expressed by the adverb "maybe", which can be used in front of all kinds of phrases, generally verb phrases.

- "The author-s of Paper42 have which name and maybe have which email?"

Negation. The negative constraint on graph patterns (`NOT EXISTS`) is expressed by the adverb "not", which can be used in front of all kinds of phrases, and in combination with auxiliary verbs. In updates, negation entails the deletion of triples.

- "Which author of Paper42 has not affiliation Salford_University?"
- "John_Smith is not a teacher and does not teach Course101."

Quantification. Quantifiers have no direct counterpart in SPARQL, and can only be expressed indirectly with negation or aggregation. In SQUALL, they are expressed by determiners like "a", "every", "no", "some", "at least 3", "the most", "the". The latter "the" is interpreted existentially in queries, and universally in updates. The universal quantifier in updates allow for batches of updates, and correspond to the use of a `WHERE` clause in SPARQL updates.

- "Every author of Paper42 has affiliation the university whose location is Salford."
- "Every author of which publication has affiliation Salford_University?"
- "Which person-s are the author of the most publication-s?"

Aggregation and grouping. Aggregation is expressed by the question determiner "how many", by relational nouns such as "number", "sum", "average", and by adjectives such as "total", "average". Grouping clauses are introduced by the word "per".

- "How many publication-s have author John_Smith?"
- "What is the number of publication-s per author?"
- "What is the average age of the author-s of Paper42?"

Expressions. Operators and functions are defined as coordinations so that they can be applied on different kinds of phrases: e.g., relational nouns, noun phrases.

- "Which publication has the lastPage - the firstPage greater than 10?"
- "Return concat(the firstname, " ", the lastname) of all author-s of Paper42."

Property paths. Property sequences and inverse properties are covered by the flexible syntax of SQUALL. Alternative and negative paths are respectively covered by the coordination "or" and the adverb "not". Reflexive and transitive closures of properties have no obvious linguistic counterpart, and are expressed by property suffixes among "?", "+", and "*".

- "Which publication-s cite+ Paper42?" (i.e., *Which publications cite Paper42 or cite a publication that cite Paper42, etc?*)

Named graphs. The `GRAPH` construct of SPARQL, which serves to restrict graph pattern solutions to a named graph, can be expressed using "in graph" as a preposition. A prepositional phrase can be inserted at any location in a sentence, and its scope is the whole sentence.

- "Who is the author of the most publication-s in graph Salford_Publications?"
- "In which graph is John_Smith the author of at least 10 publication-s?"

Graph literals. The SPARQL query forms `CONSTRUCT` and `DESCRIBE` return graphs, i.e. sets of triples, instead of sets of solutions. A `DESCRIBE` query is expressed by the imperative verb "describe" followed by a resource or a universally-quantified noun phrase. A `CONSTRUCT` query is expressed by using curly brackets to quote sentences and make them a graph literal.

- "Describe the author-s of Paper42."
- "For every person ?X that is an author of a publication that has author a person ?Y that is not ?X, return { ?X has coauthor ?Y and ?Y has coauthor ?X. }."

A detailed review of SPARQL 1.1 grammar reveals only a few missing features: (1) updates at graph level (e.g., `LOAD`, `DROP`), (2) use of results from other endpoints (e.g., `VALUES`, `SERVICE`), (3) transitive closure on complex property paths (e.g., `(^author/author)+` for co-authors of co-authors, and so on).

language	natural language	SQUALL	SPARQL
average question length	45	55	173

Table 1. Comparison of the average length of questions in the three languages.

5 Naturalness Evaluation on the QALD Challenge

The QALD[7] challenge (Query Answering over Linked Data) provides "a benchmark for comparing different approaches and systems that mediate between a user, expressing his or her information need in natural language, and semantic data". The last campaign, QALD-2, provides hundreds of questions in natural language over two datasets: DBpedia and MusicBrainz. The principle of the challenge is that a training set of 100 questions is provided, along with SPARQL translations and answers, and systems are evaluated on a test set that is made of 100 new questions. Systems are compared in terms of precision and recall for the test questions. Here, we do not measure precision and recall because SQUALL is not a system that produces answers from natural language questions, but a language that can be used to express queries. Because SQUALL has the same expressiveness as SPARQL (see Section 4), it is possible to reach perfect precision and recall. The question we try to answer in this section is: *How close to natural language questions are the SQUALL sentences, when made equivalent to the SPARQL queries?* To this purpose, we here focus on the 100 questions of the training set for the DBpedia dataset, from the QALD-2 campaign. The 100 questions of the test set are very similar, and therefore they do not add to our evaluation. The SQUALL version of those 200 questions are available from the example page of the SQUALL page. For each question, the SPARQL translation and answers from DBpedia can be obtained in two clicks.

The concision of SQUALL is comparable to natural language. Table 1 compares the average length of questions in three languages: natural language (original QALD question), SQUALL (our version of questions), SPARQL (the golden standard provided by QALD organizers). Whereas SPARQL queries are nearly four times longer than natural language questions, SQUALL queries are only about 20% longer. The difference between natural language and SQUALL is largely explained by the namespaces in qualified names (e.g., `res:IBM` instead of `IBM`).

SQUALL queries look natural. The use of variables is hardly ever necessary in SQUALL (none was used in the 100 training questions), while SPARQL queries are cluttered with many variables. No special notations were used, except for namespaces. Only grammatical words are used to provide syntax, and they are used like in natural language. There are 9 out of 100 questions where SQUALL is identical to natural language, up to proper names which are replaced by URIs:

– "Is res:Proinsulin a Protein?"

[7] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/`

- "What is the currency of res:Czech_Republic?"
- "What is the areaCode of res:Berlin ?"
- "Who is the owner of res:Universal_Studios?"
- "What are the officialLanguage-s of res:Philippines?"
- "What is the highestPlace of res:Karakoram?"
- "Give me the foaf:homepage of res:Forbes?"
- "Give me all yago:SchoolTypes."
- "Which Country has the most dbp:officialLanguages?"

Most discrepancies between natural language and SQUALL are a matter of vocabulary. Most discrepancies come from the fact that for each concept, a single word has been chosen in the DBpedia ontology, and related words are not available as URIs. Because SQUALL sentences use URIs as nouns and verbs, some reformulation is necessary. In the simplest case, it is enough to replace a word by another: e.g., "wife" vs "dbp:spouse". In other cases, a verb has to be replaced by a noun, which requires changes in the syntactic structure: e.g., "Who developed the video game World of Warcraft?" vs "Who is the developer of res:World_of_Warcraft?". An interesting example is "Who is the daughter of Bill Clinton married to?" vs "Who is the dbp:spouse of the child of res:Bill_Clinton?". The former question could be expressed in SQUALL if "marriedTo" was made an equivalent property to "dbp:spouse", and if "daughter" was made a subproperty of "child". In fact, this kind of discrepancy could be resolved, either by enriching the ontology with related words, or by preprocessing SQUALL sentences using natural words to replace them by URIs. The latter solution has already been studied as a component of existing question answering systems [3,14], and could be combined with translation from SQUALL to SPARQL.

Some discrepancies are deeper in that they exhibit conceptual differences between natural language and the ontology. We shortly discuss three cases:

- "List all episodes of the first season of the HBO television series The Sopranos!" vs "List all TelevisionEpisode-s whose series is res:The_Sopranos and whose seasonNumber is 1.". In natural language, an episode is linked to a season, which in turn is linked to a series. In DBpedia, an episode is linked to a series, on one hand, and to a season number, on the other hand. In DBpedia, a season is not an entity, but only an attribute of episodes.
- "Which caves have more than 3 entrances?" vs "Which Cave-s have an dbp:entranceCount greater than 3?". The natural question is nearly a valid sentence in SQUALL, but it assumes that each cave is linked to each of its entrances. However, DBpedia only has a property "dbp:entranceCount" from a cave to its number of entrances.
- "Which classis does the Millepede belong to?" vs "What is the dbp:classis of res:Millipede?". The natural question is again a valid SQUALL sentence (after moving 'to' at the beginning), but it assumes that `res:Millipede` is an instance of a class, which is itself an instance of `dbp:classis`. DBpedia does not define classes of classes, and therefore uses `dbp:classis` as a property from a species to its classis.

Those discrepancies are more difficult to solve. A first solution would be to make the ontology better fit usage in natural language. A second solution is to reformulate a natural question so that it matches the ontology.

6 Discussion and Conclusion

In the spectrum that goes from full natural language to formal languages like SPARQL or SQL, SQUALL (Semantic Query and Update High-Level Language) occupies a unique position. It offers the same expressiveness as SPARQL for querying and updating RDF data, and still qualifies as a controlled natural language (CNL). This means that among the natural language interfaces, SQUALL is the one that is by far the most expressive; and that among the formal languages, SQUALL is the one that is the most natural. The limit of SQUALL is that end-users have to comply with its controlled syntax, and have to know the RDF vocabulary (i.e., *Which are the classes and properties?*). However, the important result is that SQUALL can be used as a substitute for SPARQL because this entails no loss, neither in expressiveness, nor in precision.

SQUALL can be used as a front-end language when no linguistic knowledge is available about an RDF dataset, exactly like for SPARQL. As future work, SQUALL could also be used as an intermediate language, combining it with existing work in natural language interfaces. As discussed in Section 5, most discrepancies between SQUALL and spontaneous natural language are related to vocabulary and ontology. Interestingly, most of existing work have precisely focused on mapping from words to URIs and reformulation (e.g., Lemon[8]). The other way round, SQUALL provides a rich and flexible grammar (e.g., coordinations on all kinds of phrases, quantification, aggregation), and completely abstracts over low-level aspects of SPARQL (e.g., relational algebra). We therefore think that SQUALL and those existing work, while already useful individually, could strongly benefit from each other.

Future works will address (1) the full coverage of SPARQL 1.1, and its proof by implementing a translation from SPARQL to SQUALL; (2) the guided construction of SQUALL sentences with query-based faceted search [6]; and (3) the use of lexicons for more natural sentences.

References

1. Bernstein, A., Kaufmann, E., Kaiser, C.: Querying the semantic web with Ginseng: A guided input natural language search engine. In: Work. Information Technology and Systems (WITS) (2005)
2. Ceri, S., Gottlob, G., Tanca, L.: What you always wanted to know about datalog (and never dared to ask). IEEE Trans. Knowl. Data Eng. 1(1), 146–166 (1989)
3. Damljanovic, D., Agatonovic, M., Cunningham, H.: Identification of the question focus: Combining syntactic analysis and ontology-based lookup through the user interaction. In: Language Resources and Evaluation Conference (LREC). ELRA (2010)

[8] `http://lemon-model.net/index.html`

4. Dowty, D.R., Wall, R.E., Peters, S.: Introduction to Montague Semantics. D. Reidel Publishing Company (1981)
5. FerrÃ©, S.: SQUALL: a controlled natural language for querying and updating RDF graphs. In: Kuhn, T., Fuchs, N. (eds.) Controlled Natural Languages. pp. 11–25. LNCS 7427, Springer (2012)
6. FerrÃ©, S., Hermann, A.: Reconciling faceted search and query languages for the Semantic Web. Int. J. Metadata, Semantics and Ontologies 7(1), 37–54 (2012)
7. Fuchs, N.E., Kaljurand, K., Schneider, G.: Attempto Controlled English meets the challenges of knowledge representation, reasoning, interoperability and user interfaces. In: Sutcliffe, G., Goebel, R. (eds.) FLAIRS Conference. pp. 664–669. AAAI Press (2006)
8. Fuchs, N.E., Schwitter, R.: Web-annotations for humans and machines. In: Franconi, E., Kifer, M., May, W. (eds.) European Semantic Web Conference. pp. 458–472. LNCS 4519, Springer (2007)
9. Haase, P., Broekstra, J., Eberhart, A., Volz, R.: A comparison of RDF query languages. In: *et al.*, S.M. (ed.) Int. Semantic Web Conf. pp. 502–517. LNCS 3298, Springer (2004)
10. Hermann, A., FerrÃ©, S., DucassÃ©, M.: An interactive guidance process supporting consistent updates of RDFS graphs. In: ten Teije et al., A. (ed.) Int. Conf. Knowledge Engineering and Knowledge Management (EKAW). pp. 185–199. LNAI 7603, Springer (2012)
11. Hildebrand, M., van Ossenbruggen, J., Hardman, L.: /facet: A browser for heterogeneous semantic web repositories. In: *et al*, I.C. (ed.) Int. Semantic Web Conf. pp. 272–285. LNCS 4273, Springer (2006)
12. Hitzler, P., KrÃ¶tzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
13. Kaufmann, E., Bernstein, A.: Evaluating the usability of natural language query languages and interfaces to semantic web knowledge bases. J. Web Semantics 8(4), 377–393 (2010)
14. Lopez, V., Uren, V., Motta, E., Pasin, M.: Aqualog: An ontology-driven question answering system for organizational semantic intranets. Journal of Web Semantics 5(2), 72–105 (2007)
15. Montague, R.: Universal grammar. Theoria 36, 373–398 (1970)
16. Oren, E., Delbru, R., Decker, S.: Extending faceted navigation to RDF data. In: *et al*, I.C. (ed.) Int. Semantic Web Conf. pp. 559–572. LNCS 4273, Springer (2006)
17. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. In: *et al*, I.F.C. (ed.) Int. Semantic Web Conf. pp. 30–43. LNCS 4273, Springer (2006)
18. Schwitter, R., Kaljurand, K., Cregan, A., Dolbear, C., Hart, G.: A comparison of three controlled natural languages for OWL 1.1. In: Clark, K., Patel-Schneider, P.F. (eds.) Workshop on OWL: Experiences and Directions (OWLED). vol. 258. CEUR-WS (2008)
19. Smart, P.: Controlled natural languages and the semantic web. Tech. rep., School of Electronics and Computer Science University of Southampton (2008), http://eprints.ecs.soton.ac.uk/15735/